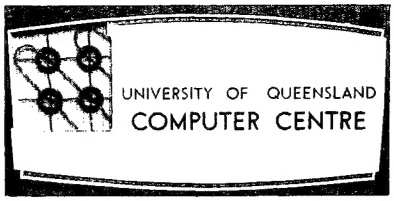

COMPUTER CENTRE BULLETIN

Vol 5 No 7
3 July 1972

Editor:
Mrs Sarah Barry

WARNING ON BASIC

[WN-89]

In some cases the diagnostic message

'INPUT DATA NOT IN CORRECT FORM - - RETYPE LINE'

is incorrectly given when only the last portion on the input line is incorrect and only that portion of the line need be retyped.

For example, if the input statement

```
20  INPUT A$,B
```

is used and no numeric data is given, by a typein of the form

```
TEST?
```

the above warning message will be given. The input to A$ of the character string 'TEST' has been satisfied. The message implies that the whole line should be retyped. In fact only the numeric data for variable B is required and that element only should be retyped.

EQUIPMENT INSTALLATION - DATA CHANNEL

[WN-93]

On Monday 5 June a second data channel was installed on the University's computer system.

Prior to this, a single data channel was utilized to connect the RD10 fixed head swapping disks, and the RP02 removable disk pack drives through a single channel to the core memory system. Considerable data traffic interference occurred between the two disk systems resulting in a large proportion of lost time, which is time lost while the processor is waiting for disk input/output to be completed. The most noticeable effect was poor response time on terminals and poor batch throughput.

With the installation of the second data channel, the RD10 fixed head disk system was connected to the second channel. This will result in a reduction of the data traffic interference between the two disk systems, and result in better performance. Initial observations of the system's performance have shown that this is the case.

FREE READ FOR PDP-10 FORTRAN (FRED)

The following article describes a free read program for PDP-10 FORTRAN written by R.A.Barham of Electrical Engineering. The editor wishes to thank Mr Barham for the program description.

This subroutine allows the reading of fully free field data from teletype, cards or disk with mixed numeric and alphanumeric (ASCII) fields and any number of fields per record. The subroutine is fully compatible with normal FORTRAN formatted reading.

The free field routine is used by writing a normal FORTRAN formatted read statement as:

READ(n_1,n_2)v_1,v_2,....,v_n

where n_1 is the unit number
n_2 is a format statement number (although the routine does not use it)
v_1,v_2,....,v_n is the variable list

Free field reading is selected by using a unit number (n) which has been previously 'enabled' for free field reading by the subroutine call:

CALL ENFRED(n_f,n_d)

where n_f is the desired free field unit number
n_d is the FORTRAN unit number of the device from which data is to be read (e.g. 5 for TTY, 10-14 for DSK).

If reading is to be from a named disk file a call to IFILE must first be made with unit number n_d.

The units given in the eight most recent calls to ENFRED are stored.

If the free field unit number (n_f) is -6 or less, only <u>one</u> record is read in (referred to later as the ONE RECORD mode). Excess variables are null or blank filled depending on whether the last field read was numeric or ASCII.

As in formatted reading a FORTRAN unit number (n_d) of -6 will cause the rereading of the previous record.

examples:

(i) CALL ENFRED (25,5) — 25 is now a free field unit for TTY input, (CDR if through

```
                                        batch)
      READ(25,10) N,(A(I)=1,N)          searches as many records as
                                        need to find all the required
                                        fields.

(ii)  CALL ENFRED(35,10)
      CALL IFILE(10,'FNAME')
      READ(35,10) N,(A(I),I=1,N)
```

Since numeric (decimal and octal), logical, and alphanumeric (ASCII) fields may be read, certain assumptions as to the type of field are made from the first character of that field. Fields are separated by spaces or commas.

Briefly, fields starting with a numeric character (digit 0-9 '.' '+' or '-') are considered numeric (see later for logical or octal fields). Other fields (even if they contain numeric characters) are considered alphanumeric fields.

Numeric fields may be in any of the standard formats acceptable to FORTRAN.

DETAILS OF FIELD TYPES

NUMERIC FIELDS

Decimal (Integer or Real)

A signed or unsigned number in standard E format.

examples:

```
25
-17
+25E1
1E1
-5.932E-07 etc.
```

The result is returned in integer or real format depending on the type of variable in the READ statement.

Octal

A signed or unsigned number preceded by a " symbol is decoded as an octal number. The number may contain up to 12 digits.

examples:

```
"52715
"-593
"-1
```

The number may contain a 'decimal' point or an E, in which case it is converted to real format IF the variable TYPE is REAL.

example:
"1E2 is read as 100 , and if the variable is REAL is converted to real format.

Numeric fields are ended by any illegal character (including excess signs and E's, and blank or comma).

LOGICAL FIELDS

A field starting with a decimal point and followed by a non numeric character (i.e. not digits 0-9 or . + or -) is assumed a logical variable. If the second character is a 'T', the variable is set to TRUE, otherwise it is set to FALSE.

examples:

(i) .T .TRUE. .TOM are read as TRUE

(ii) .F .FALSE. .XXX are read as FALSE

A logical field is ended by a space or a comma.

ALPHANUMERIC FIELDS

A string of characters, starting with a non numeric character (except " or ') and containing no spaces or commas is regarded as an ASCII field. This string is read into the variables of the READ statement in sequence (5 characters per variable for integer, real, or complex variables, and 10 characters per variable for double precision variables).

Where there are insufficient characters to fill a variable, blanks are added. Strings containing blanks or commas or starting with a numeric character can be input as ASCII by delimiting the field with apostrophes ('). Two adjacent apostrophes, insert an apostrophe (') into the text.

The following fields are read as:

ABCDEFGH	as	ABCDE	FGHbb
'52179362'		52179	362bb
ABC'EFG'		ABC'E	FG'bb

(where b represents a space).

Complex Variables

These are handled as two separate variables for numeric or for ASCII type fields (two 5 character fields for ASCII).

Double Precision Variables

These are always handled as one field (the low part is zeroed on numeric fields). Up to 10 ASCII characters can be read into a single variable (blank filled if fewer than 10 characters)

OTHER FACILITIES IN FRED

Repeated fields

An integer field followed by an asterisk (*) indicates that the following field is to be repeated.

examples:

(i) 5*21E7

(ii) 5*ABCDEF repeats ABCDEFbbbb 5 times

(iii) 5*'THIS WHOLE FIELD IS REPEATED 5 TIMES'

Skipped Variables

Adjacent commas cause variables to be skipped over without change.

example:

```
     READ(25,10) A,B,C,D,E
when reading
     7.1,5.9,,,3.1
leaves C and D unchanged (i.e. they retain the values
they had before the READ).
when reading
     5*
leaves all variables unchanged.
```

Continued Records

In the ONE RECORD mode, more than one record can be read by terminating the record with a '-', provided that character is not part of an ASCII field.

examples:

(i) 5, 7, 21, 15 - causes two records to be input

(ii) 71, 562.5

(iii) 5, 7, 21, ABC- the - sign is part of the field ABC-

Termination of Reading

In any mode an <altmode> character will cause the unsatisfied input variables to be null (or blank) filled, thus terminating the READ statement.

Use with Decode

If a call is made to the subroutine ENFDEC the NEXT DECODE statement is handled by the free field routine. The number of characters to be scanned should not exceed 140 (only 140 will be scanned if this is exceeded).

example:

```
CALL ENFDEC          enables ONE free read DECODE
DECODE (80,10,A) X,Y,Z
```

Limitations

1. Records must not exceed 140 characters in length.
2. Double precision numeric fields are handle in single precision only.

Use of FRED

FRED can be loaded from file 525.FRED as:

```
.RUN MAIN, SUB1, 525.FRED
```